TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

PROJEKT SMARTROOM

Bakalářská práce

Studijní program: B2646 – Informační technologie

Studijní obor: 1802R007 – Informační technologie

Autor práce: **Martin Brotánek**

Vedoucí práce: Ing. Karel Paleček

Liberec 2015

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

PROJECT SMARTROOM

Bachelor thesis

Study programme: B2646 – Information Technology

Study branch: 1802R007 – Information Technology

Author: **Martin Brotánek**

Supervisor: Ing. Karel Paleček

Liberec 2015

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin Brotánek**
Osobní číslo:	**M12000109**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Projekt Smartroom**
Zadávající katedra:	**Ústav informačních technologií a elektroniky**

Zásady pro vypracování:

1. Seznamte se s knihovnami PortAudio, OpenCV, Qt a dalšími pro práci se zvukem, obrazem a tvorbu grafického rozhraní v operačním systému Microsoft Windows.
2. Navrhněte inteligentní systém konferenčních hovorů. Systém by měl automaticky přenášet hlas a obraz mluvící osoby, tedy automaticky natáčet kameru a zaměřovat zvuk.
3. Systém bude vytvářen v jazyce C++ a bude na něm pracovat více lidí.

Rozsah grafických prací:	**Dle potřeby dokumentace**
Rozsah pracovní zprávy:	**cca 30 stran**
Forma zpracování bakalářské práce:	**tištěná/elektronická**

Seznam odborné literatury:

[1] **NASARRE, Jeffrey Richter and Christophe a Mark SUMMERFIELD. Windows via C/C: Prentice Hall Open Source Software Development Series. 5th ed. Redmond, Wash: Microsoft, 2011, ISBN 978-073-5663-770.**

[2] **BLANCHETTE, Jasmin a Mark SUMMERFIELD. C GUI Programming with Qt 4: Prentice Hall Open Source Software Development Series. edt. Trolltech ASA: Prentice hall, 2002, ISBN 978-0-13-235416-5.**

Vedoucí bakalářské práce:	**Ing. Karel Paleček** Ústav informačních technologií a elektroniky
Konzultant bakalářské práce:	**doc. Ing. Zbyněk Koldovský, Ph.D.** Ústav informačních technologií a elektroniky
Datum zadání bakalářské práce:	**12. září 2014**
Termín odevzdání bakalářské práce:	**15. května 2015**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci - 14 -

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
vedoucí ústavu

V Liberci dne 12. září 2014

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Mé poděkování patří Ing. Karlu Palečkovi za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování bakalářské práce věnoval.

Abstrakt

Tato práci popisuje inteligentní konferenční systém. Ten je schopný zjistit polohu mluvící osoby a otočit na ni kameru. Zároveň přenáší hlas. Systém využívá až 8 mikrofonů. Na začátku práce je specifikován hardware a zapojení. Dále následuje popis použitých knihoven. V praktické části je popsán návrh řídicí aplikace. Součástí příloh je CD s vyvinutým softwarem včetně zdrojových kódů.

Klíčová slova

Smartroom, ovládání kamery, výběr aktivního mikrofonu, konferenční hovory, konferenční místnost, DirectShow, PortAudio, Skype

Abstract

This thesis describes an inteligent conference system. It is able to detect a speaking person and focus camera on him. It also transmits voice. The system uses up to eight microphones. In the begining of this thesis there are hadware specifications and wiring. After that follows description of used libraries. In the practical part there is described the design of control aplication. As part of the annexes there is the developed software including source codes.

Keywords

Smartroom, camera control, selection of active microphone, conference calls, conference room, DirectShow, PortAudio, Skype

Obsah

Seznam zkratek

API	Application Programming Interface
MPEG	Motion Picture Experts Group
MP3	MPEG Audio Layer-3
COM	Component Object Model
GUI	Graphical User Interface (Grafické uživatelské rozhraní)
FPS	Frames Per Second (počet snímků za sekundu)
SDK	Software Development Kit
IR	Infrared (Infračervené záření)
OS	Operační systém
FIFO	First In - First Out

Úvod

Vzhledem k neustálému pokroku na poli informačních technologií se nám dostávají do rukou čím dál tím mocnější nástroje. Počítače jsou výkonnější, internetové pokrytí a jeho rychlost stoupá. I díky klesajícím cenám, které platíme za výkon, jsou technologie dostupnější širšímu okruhu zákazníků. V dnešní době se stávají trendem takzvané chytré technologie. Ty kromě vzájemného propojení zařízení s okolním světem často nabízejí pokročilejší interakci s uživatelem, či přímo inteligentní řešení nějakého problému. Doba rostoucích nadnárodních korporací si žádá řešení, která umožňují efektivní komunikaci týmů na velkou vzdálenost. Nejčastějším řešením jsou videokonference, při kterých mohou osoby i celé týmy komunikovat v reálném čase. Jakákoliv vzdálenost mezi nimi přitom nepředstavuje problém, jediné co potřebují je patřičně vybavený počítač a kvalitní internetové připojení. Právě videokonference má tato práce v plánu zdokonalit.

Při běžném konferenčním hovoru je kamera stabilně umístěna, což má za následek, že pokud se na jedné straně hovoru nachází místnost plná lidí, například konferenční místnost, má běžně uživatel na výběr pouze ze dvou scénářů. Tím prvním je, že kamera bude zabírat všechny osoby v místnosti najednou bez otáčení či přibližování mluvčích. Druhou možností je, že někdo bude kameru manuálně ovládat, tedy ji natáčet správným směrem včetně přibližování a oddalování, pokud to kamera umožňuje.

Právě druhý scénář se tato práce snaží napodobit, ovšem namísto člověka kameru ovládá řídicí program. Docílíme toho tak, že v místnosti budou rozmístěné mikrofony a kamera ovladatelná přes počítač, která by se měla automaticky otáčet

na aktivní mikrofon. Zároveň by měl být zvuk z mikrofonu přenášen spolu s obrazem přes komunikační software Skype™.

Na internetu jsou k nalezení komerční systémy nabízející podobnou funkcionalitu, ovšem jsou podmíněny zakoupením systému jako celku, tedy je nutné zakoupit hardware i software zároveň. Jsou založeny na odlišných principech, kdy většina pracuje pouze s dvěma mikrofony umístěnými v předu v místnosti. Ty zaměřují polohu zvukového zdroje v prostoru a na něj se následně natáčí kamera. Takový systém je podrobněji rozebrán v článku prof. Sayouda [10]. Oproti tomu systém v této práci navrhovaný má využívat mikrofonů rozmístěných přímo před každým mluvčím.

Inteligentní konferenční systém, který tato práce představuje, je určen pro nasazení v místnosti Smartroom, jež se nachází v prostorách Ústavu Informačních technologií a elektroniky na Technické univerzitě v Liberci.

1 Smartroom

Konferenční místnost Smartroom, nacházející se v budově 'A' Technické univerzity v Liberci, byla vybavena zařízením pro interaktivní internetové konference v programu Skype™.

Kromě stolního počítače se jedná o webovou kameru, osm mikrofonů a zvukovou kartu umožňující připojit až osm mikrofonů. V místnosti se rovněž nachází širokoúhlá televize schopná promítat obraz z počítače, rovněž tak i dataprojektor. Právě na tomto počítači lze v případě potřeby spustit řídicí aplikace, komunikační software (Skype™) a připojit k němu veškeré periferie.

1.1 Hardware

1.1.1 Kamera

Obrázek 1.1: HuddleCam-HD 10x

Pro účely konferenčního systému byla zakoupena kamera HuddleCam-HD™10x viz obr. 1.1.

Kamera disponuje servomotory umožňujícími pan a tilt neboli natočení a naklápění včetně desetinásobného optického zoomu. Je tedy umožněno pro aplikaci důležité ovládání natočení kamery ve dvou osách zároveň s přiblížením.

Obrázek 1.2: HuddleCam-HD 10x konektory

Ovládání kamery je umožněno s ní dodávaným IR ovladačem nebo přes sériovou linku (kap. 2.3) VISCA protokolem. Mimo jiné ovladač umožňuje ovládat pro účely konference důležité natočení a naklopení kamery, optický zoom a ukládání či volání aktuální pozice do paměti kamery.

Připojení je realizováno pomocí portu USB 3.0 pro přenos obrazu a RS232 pro ovládání přes počítač.[3] Konektory (obr. 1.2) se nachází na zadní straně podstavce kamery.

1.1.2 Mikrofony

Použité mikrofony jsou směrové, jednokanálové, a i když jich bylo zakoupeno celkem osm, není nutností současné využití všech.

Vzhledem k tomu, že jsou během konference staticky umístěny na stole před mluvčími, je využito přítomnosti závitu pro stativ a jsou připevněny k malým stolním stojanům.

Pro zapojení k dalším zařízením je zde využit 3pinový XLR konektor.

1.1.3 Zvuková karta

Externí zvuková karta Firestudio™Project od firmy PreSonus disponuje na čelním panelu (obr. 1.3) šesti vstupy pro mikrofony/linkové vstupy a dvěma pro mikrofony-/hudební nástroje, všechny jsou vybaveny nastavitelnými předzesilovači a vyžívají pro připojení 3pinové XLR konektory. Dále zde nalezneme stereo výstup pro 6,3 mm jack konektor.

Obrázek 1.3: PreSonus Firestudio Project - přední část

Na zadním panelu (obr. 1.4) zařízení lze najít osm obecných výstupů pro 6,3 mm jack konektor, na které lze při použití softwaru namapovat různé hudební nástroje či mikrofony s možností obohacení o efekty. Dále se zde nachází dva obecné výstupy pro 6,3 mm jack konektor pro výstup pravého a levého kanálu. Vedle nich se dále nacházejí MIDI vstupy a výstupy a stereo S/PDIF, které je zařízení schopné také využívat. V neposlední řadě je na zadním panelu obsažena dvojice firewire konektorů a vstup pro napájení.[9][8]

Obrázek 1.4: PreSonus Firestudio Project - zadní část

1.1.4 PC

Z údajů uvedených v předchozích kapitolách vyplývá, že počítač, se kterým bude systém využíván, musí disponovat hned několika vstupními / výstupními porty, bez

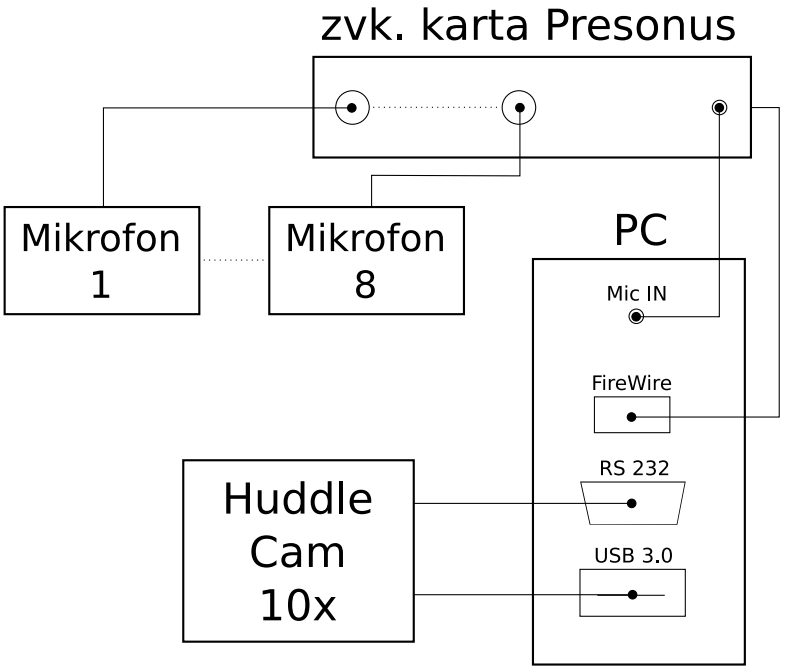
kterých by nebylo možné systém zprovoznit. Mezi ně, kromě standardních, patří zejména USB 3.0, FireWire a RS232. Dále je zapotřebí operační systém Windows od firmy Microsoft z důvodu použití DirectShow (kap. 2.2.2).

Využívaná počítačová sestava v této práci měla následující specifikace:

- Procesor: Intel® Xeon® CPU W3530 @ 2.80GHz (4 CPUs)
- Operační paměť: 8GB RAM
- Grafická karta: NVIDIA Quatro FX 1800 (4GB VRAM)
- Operační systém: Microsoft Windows 8.1 Pro

1.2 Zapojení

Jak bylo řečeno na začátku kapitoly, vše je připojeno a řízeno přes počítač. Proto jsou k němu všechny periferie připojeny viz obr. 1.5.



Obrázek 1.5: Zapojení systému

Kamera je připojena pomocí USB 3.0 k počítači pro přenos obrazu. Pro možnost jejího ovládání řídicí aplikací je dále také připojena přes sériový port. Jelikož pro její provoz není napájení přes USB dostatečné, je napájena z elektrické sítě.

V kapitole 1.1 bylo řečeno, že na přední straně zvukové karty se nachází množství vstupů pro mikrofony. K nim jsou mikrofony připojeny za použití XLR kabelů. Dále je po jejich připojení nutné tlačítkem na zvukové kartě, nacházejícím se mezi ovládacími prvky předzesilovačů, mikrofony zapnout a rovněž nastavit správné předzesílení.

Externí zvukovou kartu je rovněž nutno připojit kabelem k elektrické síti. Pro komunikaci s počítačem je zapotřebí kartu přes rozhraní firewire připojit k počítači a nainstalovat příslušné ovladače dodávané spolu se zařízením. V balení se rovněž nachází další software pro práci se zvukem, avšak jeho instalace není pro potřeby konferenčního systému nutná.

Jelikož je výsledný zvukový signál přiváděn na výstup externí zvukové karty a program Skype™pro konference vyžaduje zvuk přivedený na vstup zařízení, je nadále zapotřebí kabelem propojit mikrofonový vstup integrované zvukové karty v počítači se stereo výstupem externí karty zmíněného v kapitole 1.1.3. To lze realizovat například za použití kabelu s 3,5 mm jack konektorem na jednom konci a 6,3 mm jack konektorem na konci druhém. Případně lze také použít například 3,5 mm jack - 3,5 mm jack kabel spolu s redukcí na 6,3 mm.

2 Použité nástroje

Vzhledem k rozsahu systému, kombinaci různých technologií a použití jazyka C++ bylo zapotřebí využití několika specializovaných externích knihoven.

Rovněž výběr vývojového prostředí byl, z hlediska možného budoucího pokračování ve vývoji konferenčního systému, značně důležitý. Proto byla po pečlivé úvaze zvolena možnost vytvořit projekt na prostředí nezávislý, viz kap. 3.5.2.

2.1 Zpracování zvuku

2.1.1 PortAudio

PortAudio je volně dostupná multiplatformní audio knihovna, která pracuje pod operačními systémy Windows, Macintosh, Linux, FreeBSD, Solaris, SGI a BeOS. Poslední verze podporují množství audio systémů, jako jsou například: Windows DirectSound, Windows MME, Macintosh SoundMGR pro OS7-9 a CARRBON, Core Audio pro OSX, OSS, ASIO pro Mac a Windows, Sillicon Graphics Irix, a BeOS. [11] Tím značně usnadňuje vývoj aplikací pro různé platformy, jelikož sjednocuje odlišné frameworky pod jedno přenositelné API.

Při využíváni PortAudia je zapotřebí vytvořit callback funkci, která je volána PortAudio enginem pokaždé, když byla zachycena nějaká audio data, případně pokud jsou zapotřebí data pro výstup. [7]

Programovat je možné také přístupem blokování vstupů/výstupů. Ačkoliv to může vést k nižšímu výkonu v porovnání s callback metodou, blokování může být pro začátečníky zpočátku jednodušší k porozumění a v některých případech může být lépe kompatibilní s některými systémy. Blokování funguje v podstatě na stejném principu jako callback metoda, jen je nutné namísto volání funkce pro práci s daty poskytovat PortAudiu data v pravidelných intervalech, obvykle uvnitř smyčky.

Stream představuje aktivní proud audio dat mezi aplikací a zvukovým zařízením. Pracuje se specifickou vzorkovací frekvencí a velikostí bufferu.[1] Jelikož v počítači nelze zpracovávat nepřetržitý proud dat, jsou data postupně ukládána do takzvaných bufferů, jejichž velikost lze v PortAudiu nastavit. Obecně platí, že čím menší buffer, tím je menší zpoždění, protože jsou data dříve předána k dalšímu zpracování. Na druhou stranu to může mít negativní dopad na informační hodnotu vytěženou z bufferu, protože kratší buffer obsahuje méně dynamické informace. Proto je zapotřebí zvolit vhodnou velikost bufferu.

Při vícekanálovém zvuku je podporováno jak oddělení bufferů, kdy má každý kanál svůj vlastní, tak i prokládané buffery, kdy jsou data všech kanálů uložena do jediného bufferu. V případě prokládaného způsobu, který je v této práci využíván, jsou vždy uloženy nejprve první vzorky pro každý kanál, teprve poté následují další vzorky. Toto ukládání je pro přehlednost znázorněno na obr. 2.1, znázorňujícího uložení tříkanálového zvuku.

Alternativou k PortAudiu by mohlo být využití přímo DirectShow či knihovny SDL, nicméně vzhledem ke kvalitní dokumentaci, širokou komunitu vývojářů, množství tutoriálů a snadnou přenositelnost na jiné platformy jsem zvolil právě PortAudio.

[1] http://portaudio.com/docs/v19-doxydocs/api_overview.html

Kanál 1 vzorek 1	Kanál 2 vzorek 1	Kanál 3 vzorek 1	Kanál 1 vzorek 2	Kanál 2 vzorek 2	Kanál 3 vzorek 2	.....

Obrázek 2.1: Uložení vzorků v bufferu

2.1.2 libsndfile

Jak je uvedeno v knize Johna Halla [2], libsndfile je knihovna pro práci s audio daty v souborech, podporující značné množství formátů. Knihovnu lze použít na Unixových systémech včetně Linuxu, Mac OS X a v této práci využívaném systému Microsoft Windows. Knihovna libsndfile byla navržena a napsána Erikem de Castro Logo a je dostupná pod GNU LGPL licencí.

Alternativou by mohly být například knihovny libavcodec či Juce. Tato knihovna byla zvolena vzhledem k jednoduchosti interface, dostupnosti v předkompilované formě a široké škále podporovaných kodeků, které sjednocuje pod jedno API.

2.2 Virtuální kamera

2.2.1 OpenCV

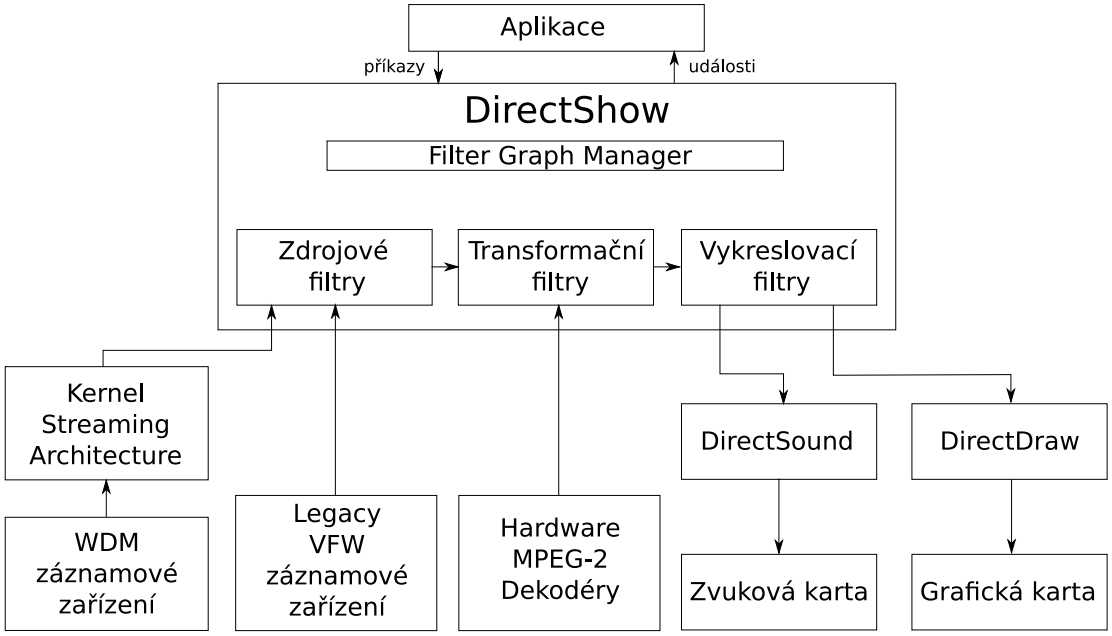
S vývojem knihovny OpenCV (Open Source Computer Vision) započal v roce 1999 Gary Bradski pod záštitou společnosti Intel pro účely aplikace počítačového vidění.

OpenCV, psané v jazyce C++, obsahuje velké množství funkcí a algoritmů využitelných pro počítačové vidění, práci s obrázky a pro nás důležitými obrazovými daty z kamery. S daty z kamery lze prostřednictvím této knihovny snadno pracovat bez nutnosti vytváření speciálních filtrů či konverze dat. Data vrací již jako uspořádanou MxNx3 matici.

Stejně jako v případě PortAudia, alternativou ke knihovně OpenCV by mohlo být přímo využití DirectShow. OpenCV je však v tuto chvíli využíváno ve velkém

množství různých aplikací a je kompatibilní s různými programovacími jazyky a platformami. Z toho důvodu jsem se rozhodl využit nástroje právě této knihovny.

2.2.2 DirectShow



Obrázek 2.2: Schéma DirectShow

Práce s médii je nesnadný úkol, protože obsahují velké množství dat, která musí být zpracována co nejrychleji a přitom zachovat audio a video stopy synchronizovány. Zároveň data mohou pocházet z různých zdrojů a v různých formátech. Programátor přitom většinou dopředu neví, jaké konkrétní hardwarové zařízení koncový uživatel použije. Právě tyto problémy se za programátory snaží DirectShow řešit. [6]

Microsoft DirectShow je architektura navržena pro jazyk C++ určená ke streamování medií pod Microsoft Windows platformou, podporující široké množství formátů. Podporuje zachycování obrazu z digitálních i analogových zařízení založených na WDM nebo Video for Windows. Podporuje hardwarovou akceleraci, je však možné využít i zařízení bez ní.

DirectShow je založeno na Component Object Modelu (COM), ovšem pro většinu aplikací není třeba implementovat vlastní objekty, DirectShow většinou potřebné komponenty již obsahuje. [5] COM komponenty jsou datové filtry, systém těchto filtrů se nazývá filter graf. Blokové schéma DirectShow viz obr. 2.2

Vzhledem k otevřenosti platformy je možné využití jakéhokoliv formátu, pokud je k dispozici filtr pro jeho dekódování. Firma Microsoft již značné množství filtrů dodává viz tab. 2.1.

DirectShow je v této práci využíváno pro vytváření virtuální kamery. Místo toho by bylo možné využít již existující hotová řešení virtuálních kamer, nicméně se mi nepodařilo nalézt žádné, které by bylo zdarma bez omezení. Většina vyzkoušených v bezplatné verzi obsahovala takzvaný *watermark*, případně nebylo podporováno FullHD rozlišení, kterým kamera (kap. 1.1.1) ve Smartroom disponuje.

2.3 Ovládání kamery

VISCA Control Protocol je protokol od firmy SONY určený pro ovládání kamer, dnes implementovaný i v kamerách jiných výrobců, přes sériové rozhraní. Definuje sadu příkazů, jako například pan, tilt, zoom a tak dále. Je založen na principu zasílání zpráv (paketů), které vypadají následovně:

- Communication speed: 9600 bps
- Start bit: 1
- Stop bit: 1
- Data bity: 8
- Parita: None
- MSB First

2.3.1 LibVisca

LibVisca je knihovna pro ovládání VISCA™ kompatibilní kamery přes RS232 port. Každá funkce v knihovně skládá dohromady až dvanáct bytů dlouhou RS232 zprávu. Zpráva je poté posílaná funkci, která nakonec pošle daných dvanáct bytů rozšířených o hlavičku a zápatí kameře. Funkce rovněž čeká na odpověď od kamery, například ACK nebo na zprávu o provedení úkonu.

Pro správnou funkčnost je zapotřebí VISCA kompatibilní kamera a počítač s operačním systémem Linux nebo Windows s RS232 portem. Implementovaná je v jazyce C. [1]

Formát	Filtr/Dekodér
AAC	Microsoft MPEG-1/DD/AAC Audio Decoder
Cinepak	
Digital Video (DV)	DV Video Decoder Filter
	DV Video Encoder Filter
H.264	Microsoft MPEG-2 Video Decoder
ISO MPEG-4 video version 1.0	
Microsoft MPEG-4 version 3	
MJPEG	MJPEG Compressor Filter
	MJPEG Decompressor Filter
MP3 (pouze dekomprese)	
MPEG-1 layer i	Microsoft MPEG-1/DD/AAC Audio Decoder
and layer II audio	MPEG-1 Audio Decoder Filter
MPEG-1 video	MPEG-1 Video Decoder Filter
	Microsoft MPEG-2 Video Decoder
MPEG-2 audio	Microsoft MPEG-2 Audio Encoder
	Microsoft MPEG-2 Encoder
MPEG-2 video	Microsoft MPEG-2 Encoder
	Microsoft MPEG-2 Video Decoder
	Microsoft MPEG-2 Video Encoder

Tabulka 2.1: DirectShow kompresní formáty

3 Řídicí program

3.1 Návrh systému

Na řídicí aplikaci bylo již od počátku kladeno mnoho požadavků. Mezi nejzásadnější patří například:

- načítání obrazu z kamery
- čtení dat ze zvukové karty
- vyhodnocení zvukových dat
- ovládání kamery
- možnost využití s komunikačními programy (Skype)

I když v tuto chvíli nijak obraz k ovládání kamery nevyužíváme, s ohledem na možný budoucí vývoj aplikace je zapotřebí předpokládat, že dojde k rozšíření o vyhodnocování obrazových dat. Pokud má být systém schopný s těmito daty získávanými z kamery pracovat a zároveň má být kamera použitelná ve Skypu, nestačí se pouze ke kameře přes systém připojit a ve Skypu ji vybrat jako zdroj obrazu. Za normálních podmínek totiž není možné, aby se k jedné kameře připojilo několik zařízení zároveň. S ohledem na tento problém bylo zapotřebí rovněž vytvořit virtuální zařízení (kap. 3.3.2), které by aktuální obrazová data obsahovalo, umožňovalo současné využití několika aplikacemi a zároveň bylo kompatibilní s komunikačními programy.

Jelikož má být systém schopen automaticky přenášet hlas a obraz mluvící osoby, je zapotřebí pracovat se zvukem. S ním souvisí, kromě samotného načítání, dvě důležité operace, kterými jsou:

- Výběr aktivního mikrofonu
- Mix kanálů

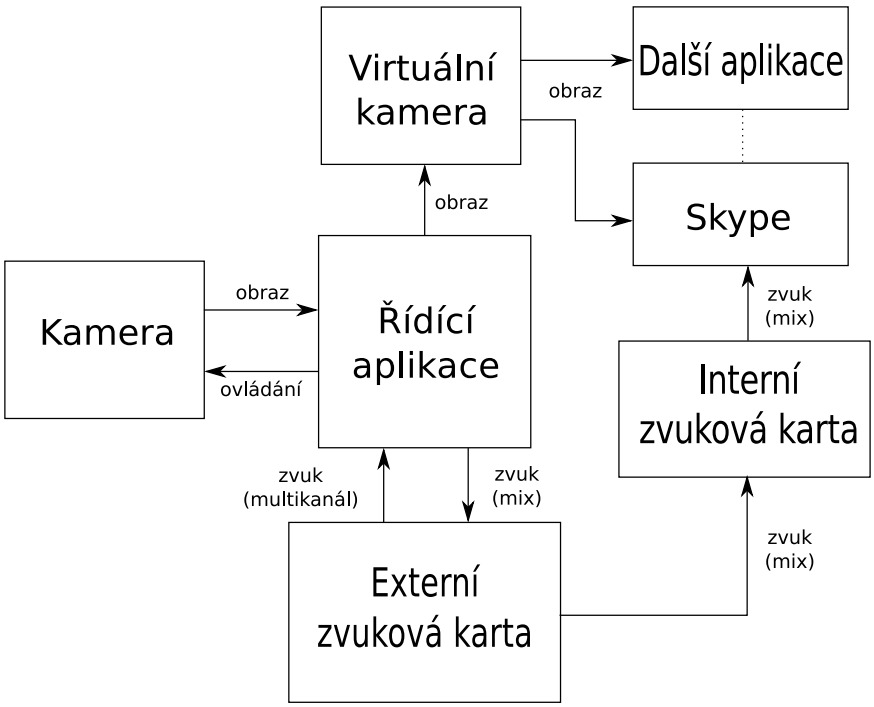
S ohledem na fakt, že jednotlivé mikrofony se na zvukové kartě reprezentují jako samostatné kanály, je zapotřebí takto k nim i přistupovat. Při zapojení osmi mikrofonů získáme zvuk o osmi kanálech, což s mnohými aplikacemi není kompatibilní. Proto je třeba redukovat počet kanálů jejich sloučením, ideálně do jednoho či dvou. Zároveň, jelikož se jedná o inteligentní systém, je vhodné neponechávat všechny kanály (mikrofony) stejně hlasité. Systém má přenášet převážně hlas aktuálně mluvící osoby, proto je vhodné nevýznamné stopy potlačit ve prospěch té, která je v dané situaci nejdůležitější. Také určení pozice pro otočení kamery je založeno na výběru mikrofonu s nejhlasitějším vstupem.

Na schématu architektury 3.1 lze pozorovat, že data z veškerých periferií jsou posílána řídicí aplikaci, která si vstupní zařízení zablokuje pro sebe. Tato data dále zpracovává a upravená předává komunikačnímu programu Skype™ v případě zvuku přes integrovanou zvukovou kartu. Video obraz kopíruje do virtuální kamery, ze které je poskytován dalším aplikacím.

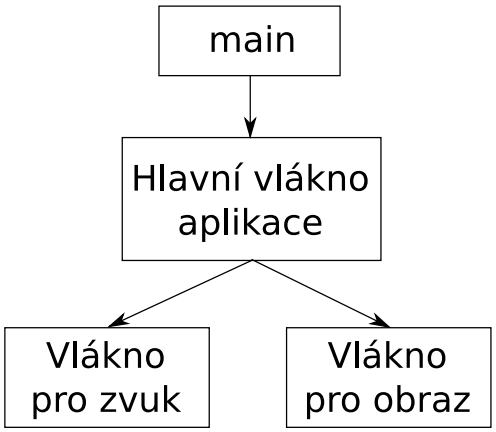
Navrhovaný konferenční systém bylo s ohledem na výše zmíněné zapotřebí navrhnout takovým způsobem, aby byl schopný ve stejný čas vykonávat hned několik různých operací. Pro takový úkol bylo zapotřebí navrhnout aplikaci, ve které budou jednotlivé dílčí úkony rozděleny do samostatně běžících vláken viz obr. 3.2

3.1.1 Komponenty

Z hlediska požadavku na dobrou přehlednost kódu, zejména s ohledem na možnost budoucího rozšiřování a vylepšování systému, byla řídicí aplikace rozčleněna do několika komponent. Jejich návrh a přehledné propojení umožňuje rychlé pochopení



Obrázek 3.1: Architektura systému



Obrázek 3.2: Rozdělení do vláken

jednotlivých částí. To je důležité, pokud by byla snaha pouze o rozšíření určité funkcionality bez nutnosti větších zásahů do ostatních částí systému. Jednotlivé funkční části mají následující zaměření:

- SmartController - hlavní třída programu

- AudioProcessor - načítání zvuku
- ChannelMixer - míchání kanálů
- CameraController - ovládání kamery
- VideoReader - čtení obrazu z kamery, zápis do virtuální kamery

Jak je zvykem, program obsahuje třídu "main", obsahující kód spouštěný při startu aplikace. V této práci má pouze jedinou úlohu, a to spustit hlavní vlákno aplikace (SmartController). Ačkoliv by bylo možné přesunout kód SmartControlleru do `main`, bylo zvoleno toto oddělené řešení s ohledem na možnost v budoucnu vytvořit grafické uživatelské rozhraní (GUI), které lze díky takto zvolenému řešení snadněji implementovat do stávající aplikace.

3.2 Vlákno pro zvuk

V kapitole 3.4 uvádím, že hlavní vlákno aplikace spouští vlákno pro zvuk. To vlákno v aplikaci reprezentuje třída `AudioProcessor` a slouží ke zpracování zvukového vstupu a vytváření zvukového výstupu pro komunikační software (Skype™).

3.2.1 AudioProcessor

Třída `AudioProcessor` pracuje se zvukovými daty, přičemž využívá knihovnu pro programovací jazyk C++ PortAudio pracující se zvukem.

Jak již bylo zmíněno (kap. 2.1.1), PortAudio může fungovat na principu blokování vstupu/výstupu nebo využívat systém callback funkcí. Právě callback funkce jsou ve vytvářeném systému uplatněny, vzhledem k jejich robustnosti a přehlednosti.

Než přikročím k podrobnějšímu popisu samotné třídy, je vhodné zmínit datovou strukturu pro ukládání audio dat, která je využívána prakticky ve všech zvuku se věnujících částech aplikace. Zmiňovanou strukturou je `AudioBuffer`, který si uchovává, mimo jiné, pro většinu algoritmů důležité:

- frames - počet rámců
- channels - počet kanálů
- data

Ukládání dat do popsané struktury zjednodušuje operace vykonávané nad daty, jelikož již většinou obsahuje veškeré potřebné informace popisující podobu dat.

Jako každá třída obsahuje i popisovaná třída konstruktor. Zde ovšem žádné zvláštní parametry nepředáváme. Po vytvoření instance jsou pro další běh důležité především metody:

- open() - otevření zvukového zařízení
- start() - start vlákna
- close() - vypnutí
- get() - získání načtených dat

V kapitole 2.1.1 je zmíněno vytváření streamu. Právě k tomu dochází v metodě `open()`. Ta má jako jediný vstupní parametr pouze ukazatel na nastavení systému získaného z konfiguračního souboru. V něm se dozví, které vstupní a výstupní zařízení plánuje uživatel využít. Systém umožňuje využít pro svou funkci libovolné zařízení, které bude svými parametry uživateli vyhovovat, není tedy limitován pouze na v tuto chvíli využívanou zvukovou kartu. Mimo to je manuální určení zařízení nutné z důvodu, že se jich v hostitelském počítači může nacházet větší množství a nelze předem odhadnout pod jakým číslem budou identifikovány. Pro výpis zařízení jsem napsal pomocný program `AudioDevices.exe`, který usnadní identifikaci v systému. Další potřebné informace pro otevření streamu se již získají z PortAudia. Jako jediný pevně nastavený parametr se vyskytuje počet kanálů výstupu, jelikož se z důvodů kompatibility výstup míchá do jediné stopy (kap. 3.2.2).

Po otevření streamu následuje volání metody `start()`, která již spustí vlastní vlákno pro zvuk, jehož kód je zapsán v metodě `run()`.

Vzhledem k využití callback funkcí se v metodě `run()` provádí pouze kontrola, zda nebyl odeslán požadavek k ukončení a ošetření výjimek, jinak stále běží zacyklený ve smyčce.

K veškeré práci se zvukem dochází v již zmiňované callback funkci. Ta má několik vstupních parametrů, ze kterých se v mém algoritmu využívají především následující ukazatele:

- *inbuf - vstup
- *outbuf - výstup
- *userData

Ukazatel `*userData` ukazuje na vytvořenou instanci `AudioProcessoru`, ve které se veškerá potřebná data ukládají do vlastních proměnných. Mezi ně patří například datová struktura `circular_buffer`, do níž ukládáme data ze vstupu zvukové karty, nacházející se na adrese `*inbuf`. Do `circular_bufferu` (což je kruhový buffer, neboli fronta typu FIFO, která je v případě naplnění od prvního prvku přepisována) se data ukládají z důvodu zamezení ztráty, kdyby k jejich zpracování nedocházelo dostatečně rychle.

Jako další dochází v callback funkci k předávání dat třídě `ChannelMixer` (3.2.2), která se stará o smíchání zvuku do jedné stopy, zároveň se zvýrazněním hlasu mluvící osoby. Po smíchání jsou data zapsána na výstup zvukové karty.

3.2.2 ChannelMixer

Třída `ChannelMixer` má za úkol z vícekanálového zvuku vytvořit jednokanálový. Toho lze docílit poměrně jednoduše, a to prostým sčítáním stop do jedné výsledné. Při takovém postupu by však nedošlo k požadovanému zesílení mikrofonu mluvící osoby vzhledem k ostatním. Aby bylo inteligentnější míchání umožněno, je zapotřebí znát:

1. Počet kanálů

2. Požadované zesílení hlavního kanálu
3. Konstanta pro zeslabené kanály
4. Konstanta pro zesílený kanál

Jelikož je třídě předáván `AudioBuffer`, informaci o počtu kanálů si již nese v sobě spolu s daty. Zesílení hlavního kanálu, respektive potlačení ostatních, si uživatel může zvolit sám a zapsat do konfiguračního souboru(4.1.1). Tato hodnota je následně při míchání kanálů zohledněna. Pokud není požadováno žádné zesílení, není tedy ani zvuk žádného mikrofonu zesilován ve výsledném mixu na úkor ostatních.

Jelikož data zvukového signálu mohou nabývat pouze hodnot v rozsahu [-1,1], při míchání více stop je zapotřebí zohlednit , aby nedocházelo k deformaci dat mimo rozsah. V nejhorším případě by mohlo dojít k sečtení vzorků, které by mohly mít maximální hodnoty, což by zaručeně v součtu mimo rozsah bylo. Pokud by nebyl požadavek na zesilování mikrofonu, stačilo by pouze každý vzorek vydělit počtem kanálů, což by celý problém vyřešilo. V nejhorším případě by tedy jejich součet byl roven jedné, což je vyhovující. Ovšem pokud má být jeden kanál silnější než ostatní, musí být i jeho vzorky dělené jinou konstantou, než u kanálů slabších. Zároveň musí být zachována podmínka, že výsledný součet nesmí přesáhnout rozsah. Z toho nám pro určení konstant vzniká soustava rovnic 3.1, kde y je hledaná konstanta pro zesílený kanál, x pro zeslabené kanály, z požadované zeslabení a N počet kanálů.

$$\begin{aligned} x(N-1)+y &= 1 \\ x/y &= z \end{aligned} \tag{3.1}$$

Po jejich vyřešení získáme vztahy 3.2 pro naprogramování výpočtu konstant v programu. Jelikož je jejich výpočet závislý na počtu kanálů a požadovaném zesílení, nelze je předem vypočítat. Proto jsou vypočítány po zavolání konstruktoru třídy, kdy již jsou veškeré potřebné informace dostupné.

$$x = \frac{z}{z(N-1)+1}$$

$$y = x/z \tag{3.2}$$

Ještě před samotným mícháním je zapotřebí vybrat nejhlasitější mikrofon, který bude ve výsledném mixu oproti ostatním zesílen. Ten určíme počítáním energie signálu, kdy nejhlasitější je ten s největší energií. S ohledem na malou velikost bufferů posílaných ke zpracování, je počítána vždy průměrná energie za několik posledních a až z nich určen nejsilnější kanál. Toto zajišťuje jisté zpoždění, kdy systém nereaguje na ojedinělý hlasitý impuls. Počet bufferů, ze kterých je počítána průměrná energie před výběrem preferovaného kanálu, lze rovněž určit v konfiguračním souboru.

Následně již dojde k sečtení kanálů, vždy s ohledem na příslušné konstanty. Výsledný mix je vrácen `AudioProcessoru`, který jej zapíše na výstup.

3.3 Vlákno pro obraz

Jelikož byl požadavek aplikaci uzpůsobit pro budoucí zpracovávání obrazových dat, vytvořil jsem vlákno pro načítání dat z kamery. Jelikož se k jedné webové kameře nemůže připojit více aplikací zároveň, k čemuž by docházelo, pokud by kameru současně využíval Skype™, bylo zároveň nutné vytvořit virtuální kameru, ke které by se Skype™mohl připojit.

3.3.1 VideoReader

Třída `VideoReader` reprezentuje samotné vlákno pro obraz. Podobně jako vlákno pro zvuk (kap. 3.2.1), i ona obsahuje metody pro otevření zařízení `open()`, spuštění vlákna `start()` a zastavení `stop()`. Současně i parametry funkce `open()` lze navolit v konfiguračním souboru. Jelikož virtuální kamera v tuto chvíli podporuje pouze jedno rozlišení, je důležité především správné nastavení zařízení pro otevření. Nicméně již v tuto chvíli je připraveno i načítání dalších parametrů ze souboru, jakožto volba rozlišení a FPS.

Stejně jako `AudioProcessor` (kap. 3.2.1), obsahuje i tato třída datovou strukturu `circular_buffer`, do které se ukládají získaná data ze zařízení. Právě z něj by v budoucnu mohla být čerpána data pro další zpracování.

V metodě `open()` dochází k otevření webové kamery, čímž si ji přisvojí a dojde k zablokování pro ostatní aplikace. Následuje přečtení jednoho snímku z kamery, ze kterého si zjistí potřebné informace o obrazu, kterými jsou velikost, výška a šířka obrázku. Na základě těchto údajů je vytvořen sdílený paměťový prostor, do kterého se data pro virtuální kameru ukládají, a do něj zapsána hlavička.

Po zavolaní metody `start()` dojde ke spuštění vlákna, jehož kód je zapsán v metodě `run()`. Zde dochází k pravidelnému přečtení snímku z kamery, který je následně zkopírován do virtuální kamery. Mimo to umožňuje obraz zároveň zobrazovat ve vlastním okně aplikace, které je otevírané knihovnou OpenCV (2.2.1).

Pro práci se sdíleným paměťovým prostorem byly využity nástroje knihoven Boost, popsaných v kap. 3.5.1. Ty celý proces z programátorského hlediska výrazně ulehčují a jsou pro většinu úkonů již dobře optimalizované.

3.3.2 Virtuální kamera

V kapitole 2.2.2 bylo řečeno, že DirectShow funguje na principu filtrů. Vytvářená virtuální kamera je ve skutečnosti *zdrojový filtr*, který se tváří jako nahrávací zařízení.

Ukázka této virtuální kamery byla nalezena na GitHubu [1] a byla rozšířena o získávání obrazových dat ze sdílené paměti, kam je zapisuje `VideoReader` (kap. 3.3.1).

Kód virtuální kamery obsahuje velké množství funkcí s různým účelem. K načtení dat ze sdílené paměti bylo využito nástrojů knihoven Boost(kap. 3.5.1).

Sdílená paměť je vytvářena `VideoReaderem`, což má za následek, že pokud neběží řídicí aplikace a virtuální kamera se pokusí z ní číst, nastane chyba, která

[1] https://github.com/diegocr/vcam

shodí celou aplikaci, která se pokusila virtuální kameru číst. To jsem ošetřil tím způsobem, že pokud sdílená paměť neexistuje, je na výstup kamery odeslán šedý obraz.

Jelikož sdílená paměť existuje dokud je k ní nějaký proces připojen, tedy například pokud z ní čte virtuální kamera, OS ji nesmaže, dokud se všichni klienti neodpojí. To může mít ten následek, že pokud je během čtení této paměti virtuální kamerou vypnuta řídicí aplikace a opět spuštěna, nemůže znovu vytvořit sdílený prostor, do kterého by opětovně zapisovala. Z toho důvodu se vždy po přečtení obrazu virtuální kamera od sdílené paměti odpojí, což má za následek okamžitý zánik paměti při ukončení řídicí aplikace. Při dalším pokusu o čtení paměti kamera zjistí, že neexistuje a na výstup odešle opět šedý obraz.

Na výstup virtuální kamery jsou data kopírována uvnitř funkce `FillBuffer`, kde zároveň dochází ke změně rozlišení obrazu. K tomu je využito funkcí knihovny OpenCV (kap. 2.2.1). V současném stavu podporuje kamera pouze rozlišení 640 x 480 pixelů, což je sice pro videokonference dostatečné, ovšem není využit plný potenciál kamery.

Pro využití virtuální kamery je nutné zaregistrovat ji v systému. To se provádí v příkazovém řádku, v administrátorském režimu, příkazem:

```
regsvr32 UMISTENI/Vcam.dll.
```

3.4 Hlavní vlákno aplikace

3.4.1 SmartController

Jak již bylo zmíněno (3.1.1), SmartController (obr. 3.3) je hlavní třída běžící v hlavním vlákně programu. Obsahuje několik základních metod pro operace se svým vlastním vláknem:

- start() - spuštění vlákna

Obrázek 3.3: Schéma hlavního vlákna

- stop() - zastavení vlákna
- run() - kód vykonávaný vláknem

Právě metody start() a stop() jsou nyní volány z "mainu", při rozšíření aplikace o GUI by stačilo pouze volat tyto dvě. Jako každá třída v objektově orientovaném programování obsahuje i SmartController konstruktor, v tomto případě se jedná o konstruktory dva. Konstruktor, který obsahuje textový řetěz jako parametr, používá daný parametr jako adresu konfiguračního souboru. Základní bezparametrový konstruktor se pokusí otevřít výchozí soubor konfigurace "settings.ini".

Značné množství algoritmů v aplikaci je závislých na určitých konstantách, jejichž nastavení má výrazný vliv na vlastnosti a chování řídicího programu. Pro možnost tato nastavení měnit bez nutnosti zásahu do kódu a následné kompilace

programu jsou tyto hodnoty načítány z konfiguračního souboru. Strukturu souboru včetně popisu jednotlivých nastavení lze nalézt v kapitole 4.1.1.

Jak již bylo zmíněno, po zavolání metody start() se spustí hlavní vlákno aplikace, jehož kód je zapsán v metodě run(). Hned prvním úkolem po startu je načíst data z konfiguračního souboru, jelikož jsou využívána ve všech následujících částech programu. Načtení probíhá ve funkci `parseOptions()` umístěné v souboru SmartController.cpp. Data jsou uložena do datové struktury "Settings", ze které jsou dále v aplikaci čtena.

Dále je zapotřebí inicializovat CameraController. Ten se stará o vlastní ovládání kamery, založeného na vyhodnocení dat ze zvukové karty.

Jako další proběhne inicializace a start vláken pro zvuk a obraz. Co se týče vlákna pro obraz, aplikace momentálně nijak obrazová data nezpracovává, nemusela by tedy být načítána vůbec. Obrazové vlákno běží z důvodů popsaných v kapitole 3.1.

Vlákno pro zvuk je již v současné konfiguraci plně využíváno. Stará se o načítání zvuku ze vstupu a zpracování zvuku pro výstup. Výstup sice není pro hlavní vlákno aplikace důležité, ovšem načtená vstupní data ano. V hlavní smyčce řídicího vlákna si právě tato data pravidelně žádá od vlákna pro zvuk a předává je CameraControlleru (3.4.2), který je následně pro své potřeby analyzuje a na základě vyhodnocených informací rozhodne o pohybu kamery.

3.4.2 CameraController

Ovládání webové kamery obstarává třída `CameraController`. V tuto chvíli umí kameru automaticky natáčet na IR ovladačem předvolené pozice na základě zvukových dat, která si sama vyhodnotí. Tato data jí jsou pravidelně poskytována z hlavního vlákna aplikace.

Třída implementuje následující metody:

- Kontruktor - předáváno nastavení, inicializace

- update() - třídě jsou poslána data, která vyhodnotí
- move() - natočí kameru na uvedenou pozici

V konstruktoru jsou předána nastavení z konfiguračního souboru (4.1.1), která si kontrolér uloží a zjistí si z nich číslo sériového portu, na kterém má s kamerou komunikovat. Poté proběhne inicializace komunikace přes VISCA protokol s využitím nástrojů knihovny LibVisca(2.3).

Ze `SmartControlleru`, popsaného v kapitole 3.4.1, jsou kontroléru pravidelně zasílaná audio data k vyhodnocení. Tato data jsou předávána metodě `update()`, která je vyhodnotí. Pokud rozhodne o natočení, zavolá metodu `move`, která natočí kameru na určenou pozici.

V kapitole 3.2.2 popisující míchání kanálů bylo řečeno, že se vybírá nehlasitější kanál počítáním průměrné energie za několik posledních bufferů. Stejný princip je využit i zde, lze ovšem zvolit odlišný počet bufferů, než při míchání kanálů.

Pro určení pozice k natočení však není pouze počítána energie a následně vybrán nejsilnější kanál, je zapotřebí zohlednit i další faktory. Mezi ně patří například situace, kdy nikdo nemluví, mikrofony ovšem detekují šum a energie je stále vypočítávána. To by bez dalších ošetření mělo za následek, že by i v relativně tiché místnosti docházelo k otáčení kamery na pozice, kde nikdo nemluví, což by postrádalo smysl. Tuto situaci jsem ošetřil tak, že pokud průměrná energie nejsilnějšího kanálu není větší než stanovený práh (určený v konfiguraci), je situace vyhodnocena jako nerozhodná.

Další problém může vzniknout, pokud je více mikrofonů zároveň podobně hlasitých. To může v některých situacích mít za následek, že průměrná vypočítaná energie bude dosahovat jen mírně odlišných hodnot a pokud situace potrvá déle, může mít každá série bufferů jinou výslednou pozici, která se bude rychle střídat mezi nejhlasitějšími mikrofony. Z toho důvodu je zjišťován nejen nejsilnější, ale i druhý nejsilnější kanál. Pokud je rozdíl jejich energií menší než stanovený práh, je situa-

ce opět vyhodnocena jako nerozhodná. To zabrání chaotickému otáčení kamery při souběžném hovoru více mluvčích.

V nerozhodných situacích se kamera natočí do výchozí pozice, kdy by měla zabírat celou místnost. Její souřadnice lze nastavit v konfiguračním souboru, pro případ, že by kamera nebyla umístěna přesně uprostřed místnosti, avšak standardně čelní pohled se nachází na souřadnicích [0,0]. K nerozhodným situacím dochází poměrně často. Například při správně nastavených prazích pokaždé, když se lidé v místnosti na okamžik utiší. Není proto vhodné, aby se kamera natočila do výchozí polohy okamžitě. Proto je nastaveno počitadlo opakujících se nerozhodných situací, až po dosažení hraniční hodnoty z konfigurace teprve dojde k návratu do výchozí polohy.

Jelikož jsou algoritmy závislé na různých hodnotách načtených z konfiguračního souboru (4.1.1), je v konstruktoru předávaný ukazatel na strukturu s konfiguračními daty.

3.5 Implementační detaily

3.5.1 Boost libraries

Boost je soubor knihoven pro programovací jazyk C++. Většina z nich je licenována pod vlastní Boost Software Licencí. Boost Software Licence je považovaná za licenci svobodného softwaru kompatibilní s GNU General Public Licence vydanou Free Software Foundation. V některých detailech se ovšem liší, například pokud jsou provedeny nějaké vlastní změny v knihovnách, autor se o ně s komunitou nemusí dělit. To umožňuje lepší utajení vlastních kódů, což může být pro komerční vývoj výhodné. [4]

Návrh knihoven umožňuje jejich využití v široké škále různých aplikací. Jejich rozsah je tak obrovský, že v některých aplikacích mohou být dostatečné bez použití dalších. Jsou stále vyvíjené a natolik rozšířené, že některé z nich již byly začleněny

do C++ standardu. Právě pro jejich široký záběr a optimalizaci jsem se rozhodl použít je v této práci a jsou využívány prakticky ve všech částech řídicí aplikace.

V této práci byly použity zejména tyto knihovny:

- Format - pro formátování textových výstupů
- Date time - pro práci s časem
- Thread - práce s vlákny
- Circullar buffer - implementující kruhové buffery
- Property Tree - datové struktury vhodné pro konfigurační data, jejich ukládání, načítánií ze souborů, atd.
- Interprocess - práce se sdílenou pamětí (důležité pro virtuální kameru)

3.5.2 CMake

CMake se používá pro přípravu projektu k překladu různými překladači. Jelikož jsem tento systém od začátku připravoval pro práci s ním, je možné pomocí něj vygenerovat projekt pro vývoj ve kterémkoliv vývojovém prostředí. Není tedy vázán pouze na Visual Studio 2013 od společnosti Microsoft, ve kterém jsem aplikaci vyvíjel já.

Umožňuje i kompilaci programu pro různé platformy, ovšem v současné verzi aplikace podporuje pouze operační systémy Windows. Je to dáno tím, že je v tuto chvíli vytvořena virtuální kamera pouze pro ně. Jelikož počítač ve Smartroom běží pod operačním systémem Windows, nebylo zapotřebí přizpůsobovat systém pro další platformy. Nicméně právě díky využití CMake by nebyl problém přeložit aplikaci pro více platforem, zapotřebí by bylo pouze připravit virtuální kamery pro další platformy a tomu přizpůsobit vlákno pracující s videem.

4 Použití systému

Během vývoje bylo využito několik volně dostupných knihoven. Ty jsou běžně distribuovány ve formě projektu pro CMake (kap. 3.5.2) se zdrojovými kódy, který je zapotřebí sestavit pro vybrané vývojové prostředí. Následně je nutná jejich kompilace a přidání do systémových cest. Stejným způsobem jsem řešil i předání zdrojových kódů této práce, kde je před sestavením projektu potřeba doplnit správné umístění knihoven v souboru `CMakeLists.txt`.

Některé knihovny (např. Boost 3.5.1) jsou dostupné i v předkompilované verzi, kdy není nutná jejich příprava pomocí CMake.

4.1 Zprovoznění systému

V počítači s běžnou softwarovou výbavou nejsou nainstalovány všechny potřebné nástroje pro chod systému. Také některé podmínky se mohou při každém použití měnit, jako například poloha mikrofonů vzhledem k umístění kamery. Proto jsou před spuštěním systému nutné následující kroky:

1. Nainstalovat všechny chybějící knihovny
2. Nainstalovat ovladače zvukové karty - např. z CD dodávaného výrobcem
3. Zapojit systém viz kap. 1.2
4. Nastavit ID zařízení v konfiguračním souboru
5. Zaregistrovat virtuální kameru v systému: `regsvr32 UMISTENI/Vcam.dll`

6. Nastavit na kameře pozice mikrofonů
7. Spustit řídicí aplikaci
8. Zahájit videokonferenci

Za předpokladu využití zkompilovaných souborů z přiloženého CD jsou pro chod systému nutné pouze knihovny PortAudio, OpenCV, LibVisca a Boost. Další jsou potřebné v případě, že bude řídicí aplikace znovu kompilována z přiložených zdrojových kódů. V tom případě je nutné doinstalovat také libsndfile a Windows Platform SDK včetně ukázkových projektů, z nichž je nutné kompilovat projekt `baseclasses`. Ten je důležitý pro virtuální kameru.

Po instalaci knihoven, ovladačů a zapojení systému je důležité správné nastavení konfiguračního souboru, jehož struktura je popsána v kap. 4.1.1. Kromě různých předvoleb je nutné nastavit hlavně ID audio a video zařízení.

Pozice mikrofonů se nastavují dálkovým ovladačem kamery. Na něm jsou tlačítka pro natáčení kamery a zoom, kterými ji lze zaměřit na požadované místo. Poté stisk tlačítka s číslem pozice následovaného tl. SET uloží souřadnice do paměti.

4.1.1 Konfigurační soubor

Prakticky ve všech částech řídicí aplikace je využíváno údajů, které může uživatel nastavit v konfiguračním souboru. Ten je načten po startu hlavního vlákna viz kap. 3.4. Nastavitelné hodnoty jsou uvedeny v tabulce 4.1. Soubor je uložen v `INI` formátu, který je snadno čitelný.

Zatímco kamera bývá identifikována pod číslem 0 v případě, že je v systému dostupná pouze jedna, u zvukových zařízení je situace komplikovanější. Jejich počet v systému bývá vysoký a je těžké určit, které číslo systém určí externí zvukové kartě. Z toho důvodu jsem napsal program `AudioDevices`, který zařízení vypíše do konzole.

Tabulka 4.1: Hodnoty v konfiguračním souboru

comport	sériový port pro ovládání
devnumber	ID kamery v systému
width	šířka obrazu (px)
height	výška obrazu (px)
fps	Počet snímků za sekundu
usevirtual	simulace otáčení kamery (true/false)
buffcount	počet bufferů pro výpočet průměrné energie
dunnobuffcount	po kolika nerozhodných situacích dojde k návratu do výchozí pozice
camdefx	X souřadnice výchozí pozice
camdefy	Y souřadnice výchozí pozice
chandiff	práh - minimální rozdíl mezi energiemi mikrofonů
noise	práh - šum
usetestfile	čtení zvuku z audio souboru (true/false)
filename	název audio souboru (testovací účely)
devnumber	ID zvukové karty - vstup
outputdev	ID zvukové karty - výstup
volumereduction	hlasitost ztišených mikrofonů (v %)
mixbuffcount	počet bufferů pro určení nejhlasitějšího mikrofonu v mixu
window_name	jméno otevíraného OpenCV okna

4.2 Simulace zařízení

Během vývoje jsem neměl vždy k dispozici veškeré hardwarové vybavení, bez kterého se správná funkčnost algoritmů dá testovat jen obtížně. Také pokud by v budoucnu měl někdo pokračovat v práci na systému a chtěl dále rozvíjet jen některé části, musel by mít k dispozici veškeré vybavení. To mě vedlo k myšlence některé zařízení simulovat.

Jako první jsem simuloval zvukový multikanálový vstup, tedy externí zvukovou kartu. Namísto ní je možné načítat zvuková data z `*.wav` souboru. Pro tyto účely jsem v systému vytvořil třídu `AudioProcessorOffline`, která kromě využití zvukového souboru funguje stejně jako pravý `AudioProcessor`. Lze tedy s její pomocí testovat otáčení kamery i míchání kanálů.

Dále bylo nasimulováno otáčení kamery, kdy namísto otočení pravé kamery dochází pouze k výpisu zpráv na konzolový výstup. Lze tedy ladit algoritmy vyhodnocující natočení kamery bez nutnosti ji mít u sebe. Pro tyto účely slouží třída `CameraControllerVirtual`.

Simulace zařízení lze spustit upravením parametrů v konfiguračním souboru (kap. 4.1.1), konkrétně parametrem `usevirtual` pro simulaci ovládání kamery a `usetestfile` pro načítání zvuku ze souboru.

5 Závěr

Cílem této bakalářské práce bylo navrhnout a realizovat inteligentní systém konferenčních hovorů. To se podařilo a systém je v tuto chvíli funkční. Dokáže tedy rozpoznat polohu mluvčího, natočit na něj kameru a přenášet jeho hlas. Zároveň umožňuje použít kameru a zvuk v programu Skype™, pro který bylo zapotřebí vyřešit problém přístupu ke kameře, kdy z ní nemohlo číst více aplikací současně. Po prozkoumání existujících řešení jsem se rozhodl tento problém odstranit vlastní virtuální kamerou. Ta v současnosti chvíli sice funguje, nabízí ovšem pouze nízké rozlišení. Zvukový výstup pro komunikační software sice v tuto chvíli systém poskytuje, není ovšem v případě zvýraznění jednoho kanálu (mikrofonu) ideální, protože zesilování/zeslabování neprobíhá zcela plynule a ve správnou dobu. Nicméně v případě, že není zvolena manipulace s hlasitostmi, poskytuje zvuk čistý a plynulý. Je tedy pro konference přijatelný. Otáčení kamery funguje v pořádku, je ovšem nutné správné nastavení prahů a citlivostí v konfiguraci. To vyžaduje práci navíc při prvních použitích systému, nicméně pokud se nemění akustické podmínky v místnosti, nemělo by být ladění konfigurace potřebné při každém použití.

Do budoucna by bylo vhodné rozšířit virtuální kameru, aby podporovala více hodnot rozlišení obrazu. Přivedení výsledného zvuku na vstup zvukové karty, aby jej bylo možné použít ve Skype™, je v tuto chvíli řešeno jack-jack zvukovým kabelem, což by bylo elegantnější provádět softwarově. Dále by při budoucí práci na systému bylo vhodné doladit ovládání hlasitostí jednotlivých kanálů ve výsledném mixu, aby byl pro lidské ucho přirozenější.

Při řešení této práce jsem se snažil používat takové nástroje, aby bylo možné vyvíjený systém bez větších zásahů přenést i na jiné platformy než OS Windows, pro který byla navržena. To se mi z velké části zdařilo, pro port na jiné platformy by bylo nutné pouze pro ně připravit virtuální kamery a upravit jejich implementaci ve třídě `VideoReader`.

Literatura

[1] Damien Douxchamps. libVISCA. [cit. 2015-04-08].
Dostupné z: http://damien.douxchamps.net/libvisca/

[2] John R Hall. *Programming Linux games*. San Francisco: No Starch Press, 2001.

[3] HuddleCam. HuddleCam-HD™10x USB PTZ Camera Manual, [on-line]. 2014 [cit. 2015-04-18].
Dostupné z: http://huddlecamhd.com/wp-content/uploads/2014/04/HuddleCam-HD-10x-User-Manual-1_1.pdf

[4] John Paul Mueller a Jeftt Cogswell. Boost Libraries and C++. [cit. 2015-04-05].
Dostupné z: http://www.dummies.com/how-to/content/boost-libraries-and-c.html

[5] Microsoft. Introduction to DirectShow, *Microsoft Development Network*. 2010 [cit. 2015-04-20].
Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/dd390351(v=vs.85).aspx

[6] Microsoft. DirectShow System Overview, *Microsoft Development Network*. [cit. 2015-04-20].
Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/dd375470(v=vs.85).aspx

[7] PortAudio. Writing a Callback Function. [cit. 2015-04-17].
Dostupné z: http://portaudio.com/docs/v19-doxydocs/writing_a_callback.html

[8] PreSonus. Features. [cit. 2015-04-05].
Dostupné z: http://www.presonus.com/products/FireStudio-Project/features

[9] PreSonus. A Versatile Solution for Small Studios. [cit. 2015-04-05].
Dostupné z: http://www.presonus.com/products/firestudio-project

[10] Halim Sayoud and Salah Khennouf and Siham Ouamour. Virtual speaker tracking by camera using a sound source localisation with two microphones. *International Journal of Networking and Virtual Organisations*, ročník vol. 12, č. issue 2, 2013: s. 85–, ISSN 1470-9503.
Dostupné z: http://www.inderscience.com/link.php?id=53733

[11] Christian Vincenot. Using portable, multi-OS sound systems. 2004 [cit. 2015-04-15].
Dostupné z: http://http://archive09.linux.com/feature/113776

Seznam obrázků